UNIVERSITÉ TOULOUSE III PAUL SABATIER

Université de Toulouse

Bureau d'étude

L3 EEA-REL

MPPT

Phaly Nathan GABET - FAYE Ousmane

Tables des Matières

Introduction

L'objectif de départ de ce bureau d'étude était de réaliser un chargeur de batterie alimenté par un panneau solaire, mais nous avons décidé de nous concentrer sur le suivit du point maximal de puissance (MPPT) du panneau solaire. Pour réaliser ce bureau d'étude, nous avons utilisé une carte Arduino qui va s'occuper de récupérer, gérer et analyser les différentes informations, afin de trouver ce point de fonctionnement.

Matériels

Pour la réalisation du projet nous disposons de plusieurs panneau solaires se situant sur le toit du bâtiment, mais pour notre système nous en utiliserons qu'un seul.
Pour la gestion du programme, nous disposons d'une carte Arduino Uno en figure 1.

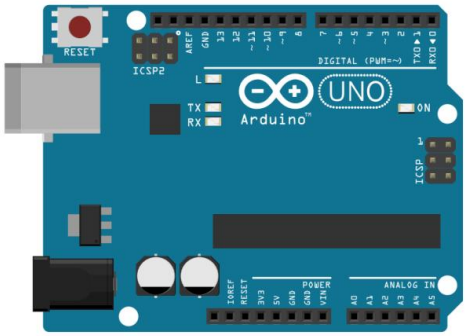

Figure [1]: Représentation graphique de la carte Arduino

Pour simplifier le câblage et pour pouvoir utiliser plus de modules nous disposons d'une extension qui se superpose à la carte Arduino, il s'agit d'une carte Shield V2 visible en figure2.

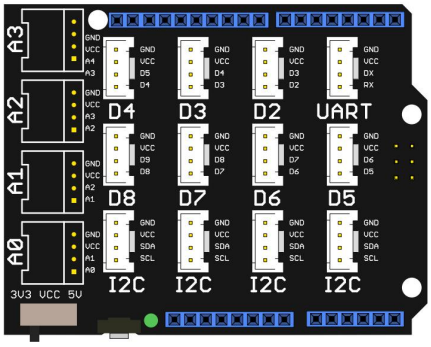

Figure [2]: Schéma de la carte Shield

Le raccordement entre les deux composants est très simple il suffit seulement de brancher le shield au dessus de la carte Arduino, car il est fait pour ne pas utiliser les port de la carte Arduino. La repésentation du câblage est visible en figure 3.

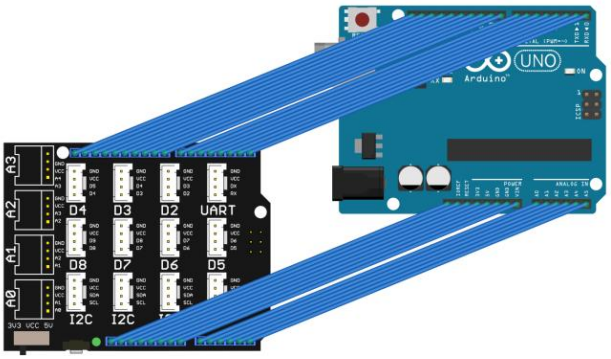

Figure [3]: Schéma de raccordement de la carte Arduino et le Shield

Pour établir le programme et la réalisation pratique du projet nous avons utilisé les documentations officielles sur la carte Arduino [1] et le Shield V2 [2].

Affichage de l’écran LCD

Dans notre système, nous avons affiché des informations en continu du panneau solaire tel que la tension et le courant maximales pouvant être fournis. Pour ce faire, nous avons utilisé un afficheur LCD qui peut indiquer 16 caractères sur 2 lignes, il affiche du texte de couleur noir sur fond jaune. L’afficheur se connecte en I2C ce qui simplifie grandement le câblage, mais nécessite d’utiliser une librairie.

Méthodes

Pour afficher du texte sur l’écran il nous faut raccorder l’afficheur à la carte Arduino, étant donné que nous avons la carte d extension Shield, nous avons simplement représenté le raccordement au niveau de l’extension. Puisque que l’afficheur se raccorde en I2C, nous utilisons que 4 broches, comme indiquées dans le tableau de la figure 4, la figure 5 représente le schéma de câblage électrique des deux modules.

Port Arduino	Broche	Couleur
GND (I2C)	GND	Noir
Vcc (I2C)	Vcc	Rouge
SDA (I2C)	SDA	Blanc
SCL(I2C)	SCL	Jaune

Figure [4]: Tableau de raccordement entre l’afficheur et le shield

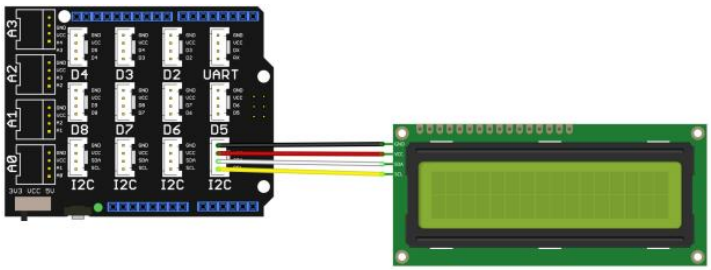

Figure [5]: Schéma de raccordement entre l’afficheur et le shield

Concernant la programmation, le principe de la communication I2C est de réduire le nombre de fils et d’utiliser un bus de communication. Nous devons donc ajouter une librairie qui a déjà été créée pour ce système d’affichage tel que la librairie «rgb_lcd» visible à la ligne 2 de la figure 6. De plus, il faut ajouter la librairie «Wire.h» qui est faite pour l’utilisation des communications en bus I2C.

```
1   #include <Wire.h>
2   #include "rgb_lcd.h"
3
4   rgb_lcd lcd;
5
6   void setup() {
7   lcd.begin(16,2);
8   }
9
10  void loop() {
11  lcd.setCursor(0,0);
12  lcd.print("Vpv:");
13  lcd.setCursor(4,0);
14  lcd.print(variable);
15  lcd.clear();
16  }
```

Figure [6]:Programme de l'affichage sur l'écran LCD

Dans le programme nous pouvons voir la fonction «lcd.» qui est la simplification de «rgb_lcd» longue à écrire. Ce paramétrage est visible ligne 4.
Pour pourvoir afficher du contenu sur l'afficheur LCD, il faut définir les caractéristiques de l'afficheur par la fonction «lcd.begin», en définissant le nombre de caractères possibles et le nombre de lignes. Il faut également placer le curseur de manière à écrire en haut de l'écran, au premier caractère. Puis avec la fonction «print», on affiche du texte en le mettant entre guillemets. Si on veut afficher la valeur d'une variable, il faut seulement mettre le nom de la variable. Pour changer le texte sur l'afficheur, il faut l'effacer, puis réaffirmer le texte ou la variable souhaitée.

Résultats

Au premier test, l'afficheur n'a indiqué que la moitié du texte souhaité. Ce problème provenait du fait que la fonction «loop» affichait puis effaçais le texte à la même fréquence que l'horloge de la carte Arduino, mais l'afficheur n'avait pas le temps d'écrire tous les caractères. Pour résoudre ce soucis il a fallu rajouter une ligne pour mettre en pause l'affichage de quelques secondes.
Une fois ce problème résolu, l'affichage est clair et le texte est suffisamment lisible, les lettres en majuscules sont facilement différentiables de celles en minuscules. Le premier programme réalisé pour afficher du texte et des variables étant long et peu pratique, nous avons choisi de créer deux fonctions externes à la fonction «void loop» simplifiant et allégeant ainsi la fonction principale, visible sur la figure7.

```
void affiche_text(const char *texte, int curseur, int ligne, int clear) {

  if (clear)
    lcd.clear();
  lcd.setCursor(curseur, ligne);
  lcd.print(texte);
}

void affiche_variable(float variable, int curseur, int ligne) {
  lcd.setCursor(curseur, ligne);
  lcd.print(variable);
}
```

Figure [7]: fonction permettant d'afficher du texte et des variables

Conclusion

L'affichage nous permet pour la suite de l'évolution du projet, de voir les valeurs reçues par les différents capteurs tel que la valeur de la tension ou du courant. Grâce aux deux fonctions d'affichage, nous pouvons facilement voir dans le programme ce qui nous intéresse.

La tension du panneau solaire

Méthodes

Pour lire la tension aux bornes du panneau solaire nous allons utiliser les ports d'entrée analogique de la carte Arduino. Dans un premier temps nous avons fait un programme de test avec une résistance variable pour vérifier si l'on réceptionnait une tension. Pour ce faire nous avons alimenté un potentiomètre qui a pour sortie l'entrée A0. Le schéma de câblage est visible sur la figure 8.

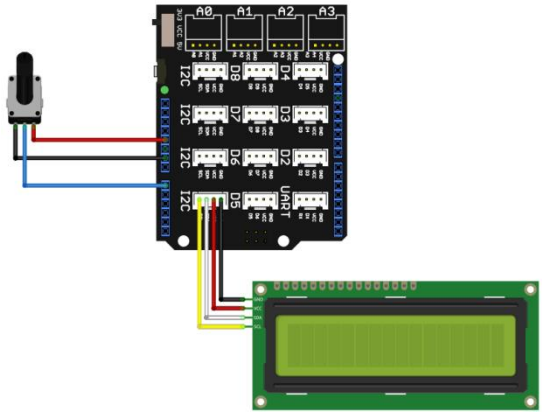

Figure [8]: Schéma de mesure de tension en A0

Le panneau solaire fournit en permanence une tension qui va nous permettre d'alimenter la batterie, pour calculer la tension maximale qui peut être produite par le panneau solaire nous avons mis en place un pont diviseur de tension.
Le panneau solaire peut fournir une tension maximale de 22V mais la carte Arduino ne peut réceptionner au maximum que 5V aux bornes de ses ports analogiques. Nous devons donc abaisser la tension pour que la tension maximale du panneau solaire soit perçue comme 5V par la carte Arduino. Ceci explique l'utilisation d'un système de pont diviseur de tension.

Résultats

Pour mettre en place ce système, il faut utiliser la formule est Vs=Vcc*(R2/(R1+R2)).Où R1 est la résistance en amont du système alimenté par la tension Vcc et R2 celle reliée à la masse. Entre les deux résistances, nous tirons un fil jusqu'à l'entrée A0 de la carte Arduino. Dans notre cas, nous avons une tension Vcc de 22V et une tension Vs qui doit être de 5V. Il nous faut donc un rapport de réduction de 0.227. Nous choisissons une résistance de valeur de 10kΩ et par le calcul nous trouvons qu'il faut une résistance de 34kΩ pour R2. N'ayant pas de résistance de cette valeur, nous optons pour deux résistances en série de 12kΩ et 22kΩ. En figure 9, nous avons le schéma électrique du pont diviseur de tension raccordé au Shield de la carte Arduino, la valeur de la tension est affichée sur l'écran LCD.

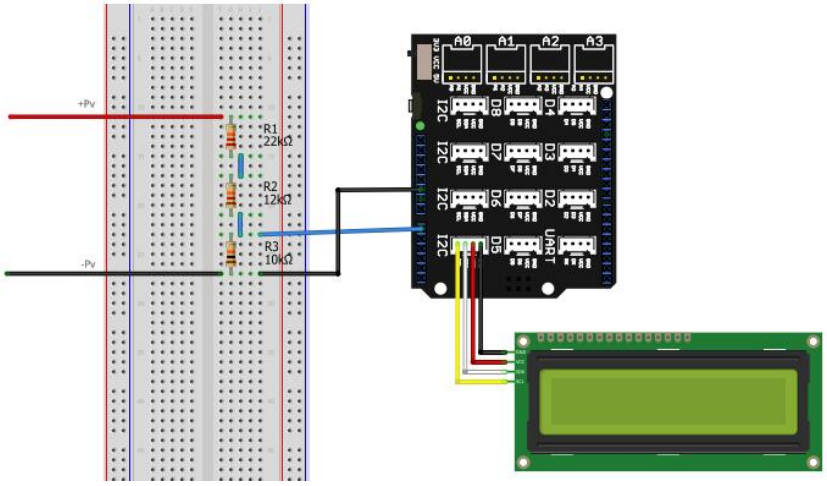

Figure [9] : Schéma du pont diviseur de tension du panneau solaire

Conclusion

Après un raccordement au panneau solaire nous avons effectué un programme qui permet d'afficher la tension reçue en volt directement sur l'afficheur LCD, en prenant soin d'annuler le rapport de réduction, afin d'avoir la vraie valeur du Panneau Solaire.

Le courant du panneau solaire

Méthodes

Pour la mesure du courant nous utilisons un capteur de courant, ce dernier nous permet de brancher d'un côté le panneau solaire et de l'autre de le raccorder à la carte Arduino par des fils. Ce capteur fonctionne grâce à une communication en Bus SPI, il nous faut donc effectuer une récupération des données. Pour ce faire nous devons alimenter le capteur en 3.3V et raccorder une masse GND comme indiquée sur la documentation du constructeur [4] et raccorder les trois pins de communication qui sont CS,CLK et MISO (comme en figure 10). Le schéma de câblage est visible en figure 11.

Port Arduino	Broche	Couleur
+3.3V	Vcc	Rouge
GND	GND	Noir
10	Chip Select	Vert
13	Clock	Jaune
12	MISO	Blanc

Figure [10]: Tableau de câblage du capteur de courant

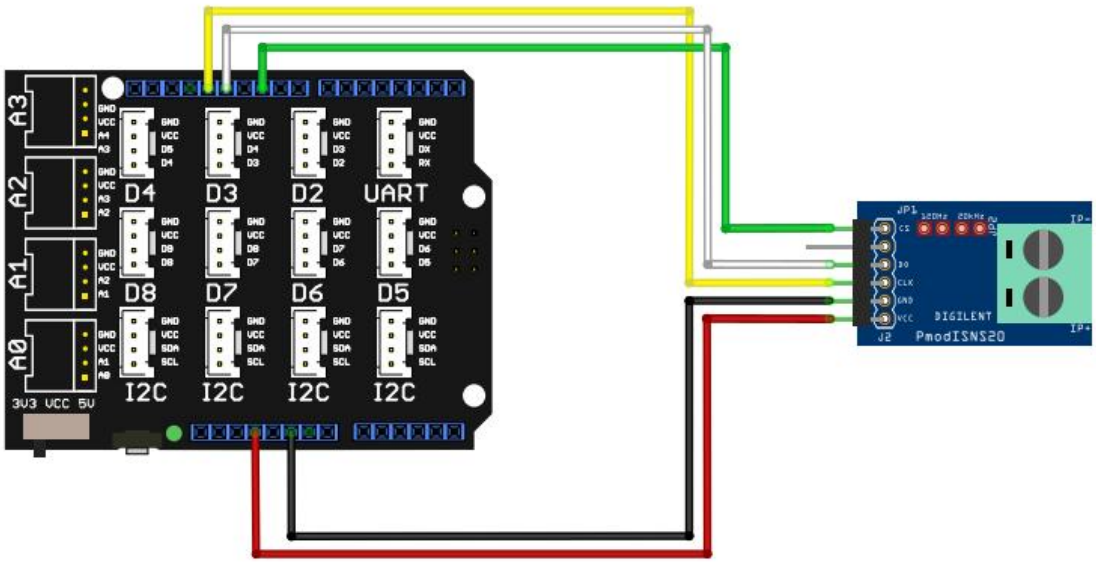

Figure [11]Schéma de raccordement entre le capteur de courant et le shield

Le port CS signifie *Chip Select*. On va l'utiliser lorsque l'on souhaite lire les données du capteur. Pour celui-ci, il faut lire les données lorsque la valeur du CS est à 0V. Nous allons donc câbler le chip select sur le port 10 de la carte Arduino. En ce qui concerne le programme, pour définir le fait que nous allons lire et enregistrer les valeurs seulement à l'état bas du CS, nous allons, dans un premier temps, dire que nous avons le pin du *Chip Select* sur le port 10 de la carte Arduino,

comme indiqué en figure 12.

```
#define CS 10              // affectation de la broche CS
```

Figure [12]: Affectation de la broche 10 à la variable CS

Dans un second temps, après avoir préciser que CS était une sortie, nous la mettons à l'état bas pour lire et enregistrer les valeurs dans les variables MSB et LSB. Ensuite, nous remettons le CS à l'état haut pour cesser de lire les valeurs. Cette partie du programme est visible en figure 13.

```
digitalWrite(CS, LOW);     // activation de la ligne CS
MSB = SPI.transfer(0x00);  // récupération des bit de poids forts
LSB = SPI.transfer(0x00);  // récupération des bit de poids faibles
digitalWrite(CS, HIGH);    // désactivation de la ligne CS
```

Figure [13]: Mise à l'état bas puis haut du *Chip Select*

Le CLK est l'abréviation de *clock* qui est l'horloge, il s'agit de la fréquence à laquelle on va pouvoir lire un bit, plus la fréquence est élevée, plus les impulsions vont être proches et plus nous pourrons lire de valeurs rapidement. Nous avons décidé d'utiliser une fréquence de communication pouvant être générée par la carte Arduino. Nous avons donc choisi de fonctionner sur une fréquence de 1MHz pour être sur de pouvoir récupérer toutes les informations du capteur, il faut donc paramétrer le fonctionnement SPI de la carte Arduino. On commence par activer le mode SPI de la carte Arduino, étant donné que l'on doit lire les valeurs du capteur, quand le CS est à l'état bas on doit se mettre en mode0 du SPI et on indique la fréquence de l'horloge qui est pour nous à 1MHz. On peut voir le paramétrage du mode SPI en figure 14.

```
SPI.begin();
SPI.setDataMode(SPI_MODE0);
SPI.setClockDivider(SPI_CLOCK_DIV16);
```

Figure [14]: paramétrage du mode SPI de la carte Arduino

Le MISO signifie *Master In Slave Out*, c'est sur ce port que nous allons récupérer les données qui seront sous forme de bit à l'état 1 ou 0 selon la valeur du courant, ils seront synchronisés à la fréquence de l'horloge ce qui nous permettra de lire les bits du signal. Les données du capteurs sont codés sur 12 bits avec un bit de signe, cela veut dire que le bit 2^{12} est à 1 si le courant est positif et à 0 s'il est négatif, les autres bits permettent de calculer la valeur du courant.
Le MSB signifie *most significative bit ou* bit de poids fort, et LSB est son inverse *Less significative bit* donc bit de poids faible. Ces deux variables contiennent 8 bits de données et nous allons les rassembler pour former un Mo qui contiendra la valeur du courant. Puis de cette valeur, nous appliquerons la formule de la figure 15 qui est donnée par le constructeur [4] pour convertir cette valeur en courant.

$$I_{mA} = \frac{1000}{89.95}(ADC_{VALUE} - 2048)$$

Figure [15]: Formule du courant donnée par le constructeur

Nous enregistrons le résultat de ce calcul dans une variable que l'on peut afficher sur l'écran LCD. Nous pouvons voir en figure 16, la variable qui enregistre le calcul du courant

```
milli_amps = 1.1335 * (1000 / 89.95) * (((MSB << 8) | LSB) - 2048);
```

Figure [16]: calcul du courant à partir du MSB et du LSB

Résultats

Après avoir effectué et téléversé le programme, nous avons mis un oscilloscope en sortie des pins afin de récupérer les différents signaux électriques entant et sortant du capteur de courant. Une fois les broches Chip Select, MISO et CLK branchées à l'oscilloscope nous pouvons visualiser la figure 17.

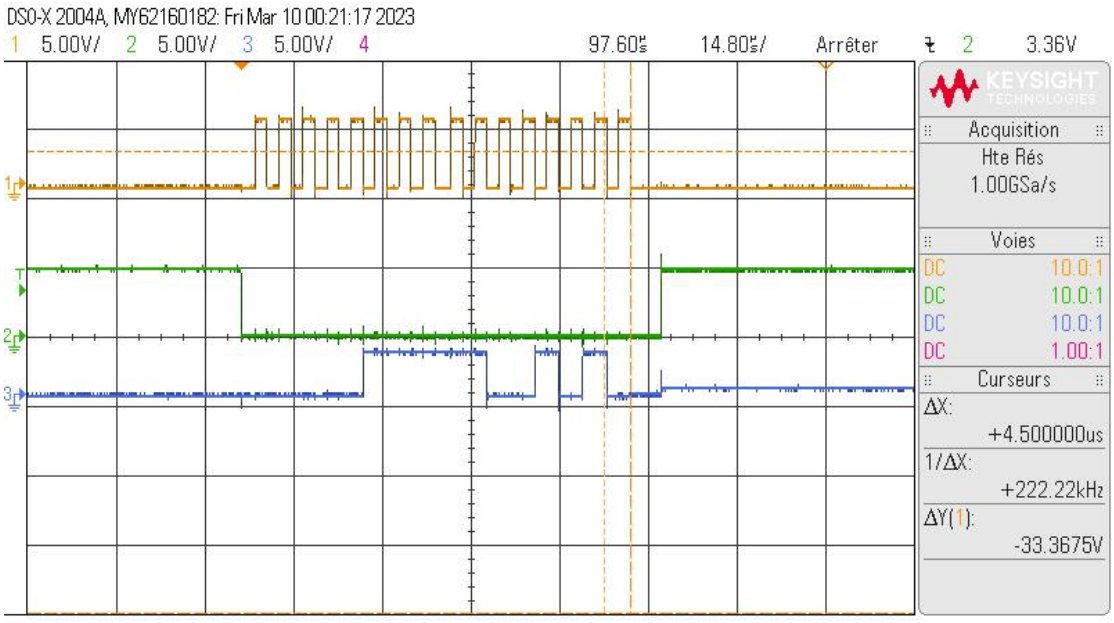

Figure [17]: Visualisation des broches CS, MISO et CLK à l'oscilloscope

Nous pouvons voir en orange le signal de l'horloge ainsi que les 16 fronts d'horloge. Le signal vert est celui du *Chip Select*, on peut noter qu'il n'y a pas de front d'horloge lorsque qu'il est à l'état haut. Le signal bleu est celui du signal du MISO et donc lorsque l'on compte le nombre de bits à l'état haut sur le MISO, on peut lire la valeur du courant, dans ce cas là, d'après le capteur le courant vaut 2.5A.

Nous avons donc décidé de faire plusieurs mesures de courant pour vérifier la précision du capteur. En disposant un ampèremètre comme valeur de référence et le capteur de courant en série nous pourrons voir si le courant calculé par le capteur est bien le même que celui de l'ampèremètre.
La figure 18 prise grâce à un oscilloscope PICO représente le signal MISO du capteur de courant aligné au signal de l'horloge.

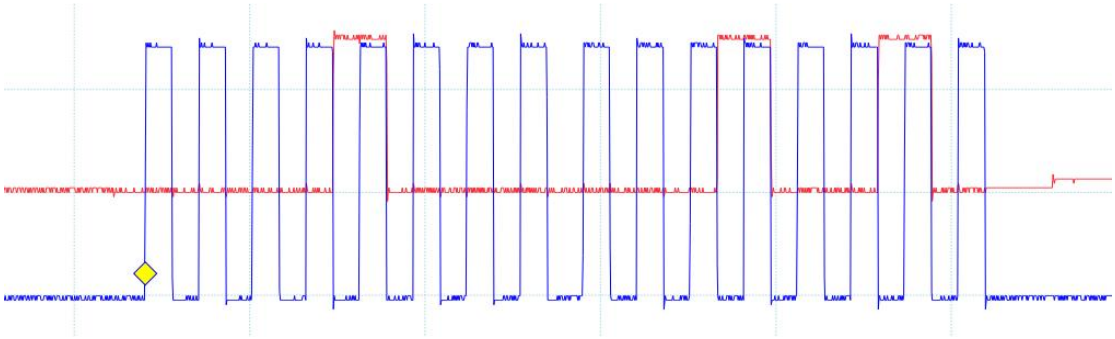

Figure [18]: Signal d'horloge et signal MISO du capteur de courant

Pour la lecture des bits nous regardons si le signal du MISO est à l'état 1 en même temps que le front montant de l'horloge. Nous pouvons voir que le bit 12 est à l'état 1 ce qui signifie que le courant est positif. Parmi les bits qui permettent de calculer la valeur du courant nous pouvons voir que les bits 2 et 5 sont également à l'état haut. Lorsque l'on fait la somme des bits nous trouvons une valeur de 2066. En appliquant la formule du constructeur nous avons une valeur de courant de 0.20A alors que la valeur réelle est de 0.28A ce qui nous fait une erreur de 28.6%.
Nous avons refait des mesures avec un courant plus élevé pour voir si la marge d'erreur variait ou non. En figure 19 nous avons les signaux dans le cas où le courant vaux 1.05A.

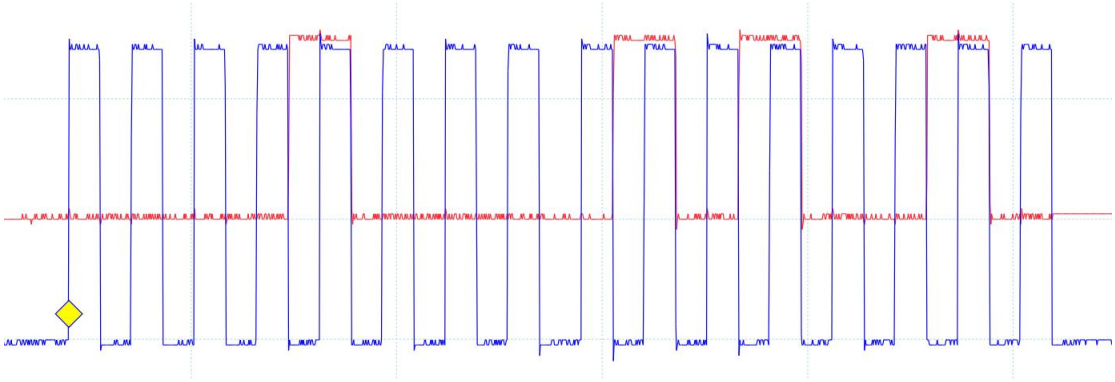

Figure [19]: Signal d'horloge et du MISO à 1.05A

Avec ce deuxième exemple, après calcul nous avons une valeur de 0.911A au lieu de trouver 1.05A ce qui nous fait une erreur de 13%.

Conclusion

Grâce au test nous pouvons voir qu'il y a une marge d'erreur qui est de 28%, nous avons donc choisi d'appliquer une correction afin de réduire ce pourcentage d'erreur à 2% en ajoutant un facteur d'amplification de 0.132%. Après correction, nous avons pu mettre en place le tableau de la figure 20 qui représente les différentes valeurs du courant avec et sans la correction.

Courant réel	Courant du capteur sans correction	Courant du capteur avec correction
0 A	0 A	0 A
0,5 A	0,35 A	0,4 A
1 A	0,8 A	0,93 A
1,5 A	1,27 A	1,43 A
2 A	1,71 A	1,94 A
2,5 A	2,17 A	2,44 A

Figure [20]: Tableau des différentes prises de mesures du courant

La différence entre la valeur réelle est la valeur mesurée sur un courant inférieur à 0.5A est assez importante car le capteur a une marge d'erreur de 2% comme indiqué sur la datasheet, mais cela ne nous dérange pas, car nous travaillons sur des valeurs comprises entre 1 A et 6A.

Commande du hacheur

Introduction

Pour commander le hacheur, nous allons utiliser la gâchette du transistor qui se trouve dans le circuit du hacheur. Il faut le commander à une fréquence minimale de 20kHz, mais c'est une fréquence audible de ce fait nous avons décidé de l'utiliser à 50kHz. Pour effectuer la commande du hacheur nous allons utiliser la fonction PWM de la carte Arduino, seulement, la sortie de la carte fournit une tension de 5V or la gâchette du transistor s'active à une tension minimale de 10V. Nous allons donc mettre en sortie du PWM un Amplificateur Opérationnel.

Partie 1 : PWM

Introduction

La fonction PWM, *Pulse Width Modulation*, de la carte Arduino signifie Modulation de largeur d'impulsion, elle permet de créer une signal périodique d'une fréquence de 500Hz, dont la largeur de l'impulsion peut être modifiée ainsi que le choix de port de sortie du signal.
Dans notre cas nous allons modifier la largeur de l'impulsion mais aussi travailler sur la fréquence plus élevée, pour cela nous devons faire des modifications dans les registres afin d'élever la fréquence de sortie. Ensuite il faut augmenter la fréquence de l'horloge des sorties PWM et donc nous allons faire des modifications sur le compteur de la carte Arduino afin de l'accélérer et donc de pouvoir atteindre cette fréquence, mais cela ne peut s'appliquer qu'à certaines sorties appelées Timers, qui comporte deux sorties, port A et port B. En regardant le principe de fonctionnement des Timers de la carte Arduino [5], nous avons décidé de travailler avec la Pin 3 qui correspond au port B du Timer 2 .

Méthode

Au niveau du programme, il faut définir la pin 3 comme sortie, et paramétrer les différents registres afin d'avoir les résultats souhaités. Selon la documentation de l'ATMega328P [6] il nous faut régler la sortie en *Fast PWM Mode*, ce mode permet d'avoir une fréquence de sortie la plus élevée possible. Nous devons choisir dans un premier temps la fréquence voulue grace à l'équation de la figure 21.

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \times 256}$$

Figure [21]: Equation permettant de trouver la fréquence de sortie

Dans cette équation «N» est la variable de *prescaler*, en attribuant une valeur à cette variable, nous pouvons modifier la fréquence de l'horloge interne. Le problème c'est qu'elle ne peut prendre que les valeurs 1, 8, 64, 256 ou 1024. Avec les différentes valeurs possibles nous ne pouvons pas avoir le fréquence de 50kHz souhaité, nous devons donc trouver une alternative.
L'autre moyen d'avoir la fréquence du PWM souhaitée est de créer un signal périodique et d'activer la sortie à chaque fois que la valeur souhaitée est dépassée. Selon le signal périodique choisit nous pouvons avoir une fréquence très élevée ou avoir une fréquence peu élevée mais plus précise. Pour cela il faut régler plusieurs registres.

Les registres

Les registres se composent de plusieurs bits que l'on doit activer ou pas selon le mode de fonctionnement souhaité. Pour les activer lors de la programmation, on peut écrire manuellement le nom des bits que nous voulons mettre à 1 ou alors on peut donner à la variable correspondant au registre une suite de 0 et de 1 qui signifie si le bit doit être à l'état 1 ou 0. Nous choisissons le deuxième système de paramétrage car, pour certain cas, il nous est nécessaire de déclarer explicitement l'état de la variable.

TCCR2A

Le premier registre à paramétrer est le TCCR2A qui signifie *Timer/Counter Control Register2 A*. Il nous permet de régler l'horloge de fonctionnement de la sortie PWM. Le paramétrage du registre se fait grâce aux 6 bits pouvant être mis à l'état 1 (activé) ou 0 (désactivé) visibles en figure 22.

TCCR2A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
(0xB0)	COM2A**1**	COM2A**0**	COM2B**1**	COM2B**0**	–	–	**WGM21**	**WGM20**	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure [22]: Bits du registre TCCR2A

Les bits COM2A et COM2B vont permettre de choisir la manière dont on souhaite utiliser et les ports OC2A et OC2B qui sont les ports dee sorties qui s'activent lorsque l'un des bit COM2 est activé. Le paramétrage des ports selon l'utilisation que l'on veut en faire, sont visibles en figure 23 et figure 24 .

Table 17-3. Compare Output Mode, Fast PWM Mode(1)

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal port operation, OC0A disconnected. WGM22 = 1: Toggle OC2A on compare match.
1	0	Clear OC2A on compare match, set OC2A at BOTTOM, (non-inverting mode).
1	1	Set OC2A on compare match, clear OC2A at BOTTOM, (inverting mode).

Figure [23]: Modes de fonctionnent de COM2A1 et COM2A0

Table 17-6. Compare Output Mode, Fast PWM Mode(1)

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected.
0	1	Reserved
1	0	Clear OC2B on compare match, set OC2B at BOTTOM, (non-inverting mode).
1	1	Set OC2B on compare match, clear OC2B at BOTTOM, (inverting mode).

Figure [24]: Modes de fonctionnement de COM2B1 et COM2B0

Dans notre cas nous travaillons que sur le Pin 3 qui correspond au OC2B. Nous souhaitons que la valeur soit à l'état haut, quand le compteur a atteint la valeur maximale, donc nous choisissons le mode non inversé.

Les deux bits WGM21, WGM20 fonctionnent avec le troisième WGM22 (dans les registre TCCR2B), ils nous permettent de choisir le mode de fonctionnement du PWM visible en figure 25.

Table 17-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Figure [25]: Paramétrage du mode de fonctionnement

Dans notre cas nous voulons une fréquence spécifique qui est très élevée, nous allons donc nous mettre en mode *Fast PWM* et pour avoir une fréquence précise nous allons comparer deux valeurs et activer la sortie dès que le compteur aura atteint la valeur maximale. Celle-ci sera OCRA (TOP) et nous remettrons à zéro (BOTTOM) le compteur dès qu'il aura atteint la valeur maximale.

TCCR2B

Le second registre à paramétrer est le registre TCCR2B qui signifie *Timer/Counter Control Register 2B*, il contient également 6 bits pouvant être réglés.Ils sont visibles en figure 26.

17.11.2 TCCR2B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure [26]: Bits du registre TCCR2B

Les bits FOC2A et FOC2B ne sont utilisés que lorsque l'on est pas en mode PWM ce qui n'est pas notre cas et dans la documentation technique, il est stipulé qu'il faut mettre le bit à 0 si on ne veut pas l'activer.

Nous retrouvons le bit WGM22 permettant le paramétrage du mode de fonctionnement.

Les bits CS22, CS21 et CS20 permettent de donner une valeur de *prescaler* dans le cas où on en utilise un. Le tableau de paramétrage est visible en figure 27.

Table 17-9. Clock Select Bit Description

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(no prescaling)
0	1	0	clk_{T2S}/8 (from prescaler)
0	1	1	clk_{T2S}/32 (from prescaler)
1	0	0	clk_{T2S}/64 (from prescaler)
1	0	1	clk_{T2S}/128 (from prescaler)
1	1	0	clk_{T2S}/256 (from prescaler)
1	1	1	clk_{T2S}/1024 (from prescaler)

Figure [27]: Tableau de paramétrage des bits CS22, CS21 et CS22

Nous allons travailler dans le troisième mode, c'est à dire avec un *prescaler* de 8 . La raison sera expliqué dans la partie suivante.

Résultats

Dans le programme le paramétrage de la fréquence se fera qu'une seule fois et c'est le rapport de cycle que nous allons modifier afin de pouvoir bénéficier de la puissance maximale du panneau solaire.

Nous avons choisi d'émettre la fréquence sur le port 3 car il était modifiable par un Timer et que nous ne l'utilisions pas précédemment. De ce fait nous devons le déclarer comme port de sortie grâce à la fonction pinMode de la carte Arduino, à partir de la nous pouvons paramétrer les registres TCCR2A et TCCR2B et une fois que c'est fini, nous pouvons régler les valeurs minimales et maximales des variables OCR2B et OCR2A.

La valeur de OCR2A va nous permettre d'avoir la fréquence de 50kHz.
Auparavant, grâce à la formule de la figure 21, on a vu que l'on ne pouvais pas juste appliquer un *prescaler* car la fréquence en était alors trop basse ou trop élevée. Il nous faut donc étudier le fonctionnement du signal du PWM. Sur la figure 28 suivante nous pouvons voir le signal périodique qui va nous permettre d'atteindre une fréquence de 50kHz.

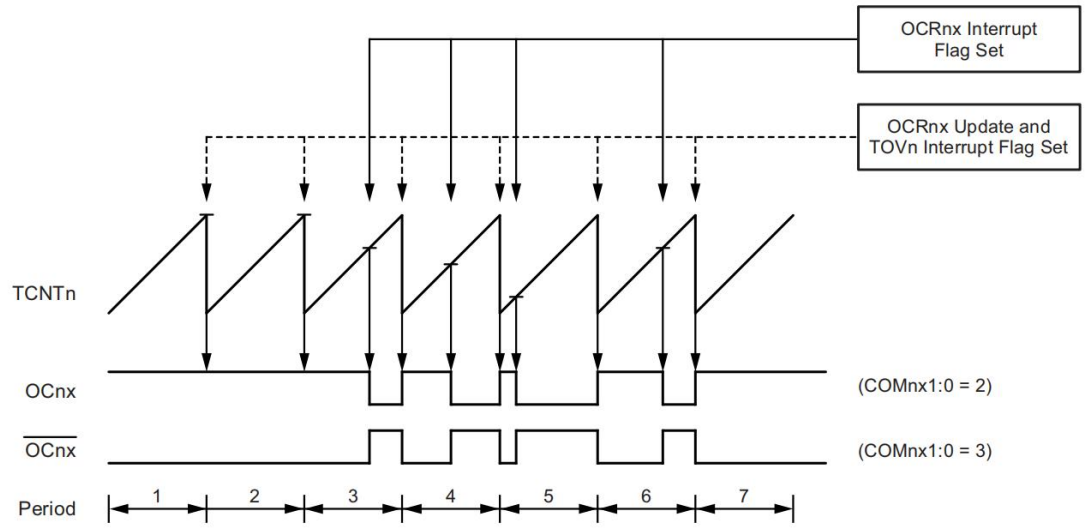

Figure [28] : Signal d'horloge du PWM

On peut voir dans la figure 28, le signal TCNTn est le signal périodique du PWM. Dans notre cas le Timer 2 fonctionne sur 8bits, donc, a un plage de 0 à 255. Nous pouvons voir qu'il y a des flèches en trait pleins qui indique *OCRnx Interrupt Flag Set*, ce paramètre correspond à notre OCR2A. Il s'agit de la valeur maximale d'incrémentation que peut prendre le compteur avant de se mettre à 0. Plus cette valeur est basse, moins le compteur va mettre de temps à s'incrémenter et plus la fréquence sera élevée. Si nous n'appliquons pas de *prescaler* le fréquence minimale que l'on va pouvoir avoir sera de 62500Hz, ce qui est déjà trop. Nous appliquons donc un *prescaler* de 8 pour avoir une fréquence minimale de 7812Hz. Nous allons alors calculer la valeur qu'il faut donner à OCR2A.
Une fréquence de 7812Hz correspond à une période de 128µs, et une fréquence de 50kHz correspond à une période de 20µs. Par un calcul simple on en déduit que l'on peut découper la période de 128µs en 6.4 périodes de 20µs. Notre signal PWM a une amplitude de 255, nous allons donc diviser le valeur maximale par 6.4, ce qui nous permettra de savoir quel valeur il faudra donner à OCR2A pour avoir la bonne fréquence. On en déduit donc qu'il faut une incrémentation maximale jusqu'à 39 pour avoir une fréquence de 50kHz. La variable OCR2B correspond au *duty cycle*, ou rapport de cycle, lui s'étendra entre 0 et 100%.

Conclusion

En figure 29, nous faisons le paramétrage de la fonction PWM à 50kHz.

```
pinMode(3, OUTPUT);
TCCR2A = 0b00100011;
TCCR2B = 0b00001010;
OCR2A = 39;
OCR2B = 0;
```

Figure [29] : Paramétrage de la fréquence de 50kHz

Les deux premiers bits du registre TCCR2A sont à 0 car on n'utilise pas cette sortie, les deux suivants sont respectivement à 1 et 0, nous permettant de mettre la deuxième sortie du timer (Pin 3) à l'état haut, quand le compteur a atteint la valeur souhaitée. Les deux derniers bits et le cinquième bit du second registre sont à 1, car ils indiquent que nous fonctionnons en mode fast PWM et que la sortie OCR2A est la valeur haute. L'avant dernier bit du second registre est à l'état 1, car il nous permet d'appliquer un *prescaler* de 8. Et on applique à OCR2A une valeur de 39 afin d'avoir la fréquence exacte.

Partie 2 : Amplificateur

Pour faire passer une tension de 5V à 10V il nous faut la multiplier par deux, nous allons donc utiliser un Amplificateur Opérationnel Parfait (AOP) en amplificateur non-inverseur, le câblage à mettre en place est visible en figure 30, sa fonction de transfert est la suivante : $Vs = (1 + \frac{R2}{R1})Ve$

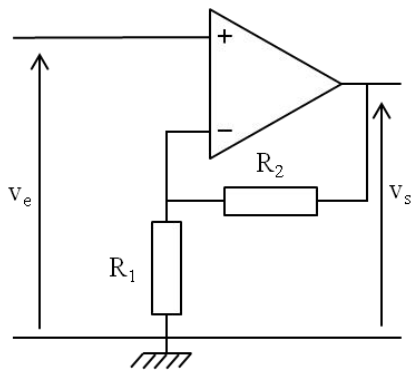

Figure [30]: Schéma électrique d'un Amplificateur non-inverseur

Avec Ve la tension d'entrée et Vs la tension de sortie, de plus pour avoir un gain d'amplification de deux, il nous faut donner aux résistances les mêmes valeurs.

Méthode

Pour la réalisation, nous allons utiliser l'amplificateur TL071, ce composant comporte un seul Amplificateur Opérationnel ainsi qu' une bande passante de 3MHz et peut être alimenté par des tensions de +15V et -15V, ses broches de raccordement sont 7 et 4, le signal de sortie est récupéré sur la pin 6, visible en figure 31.

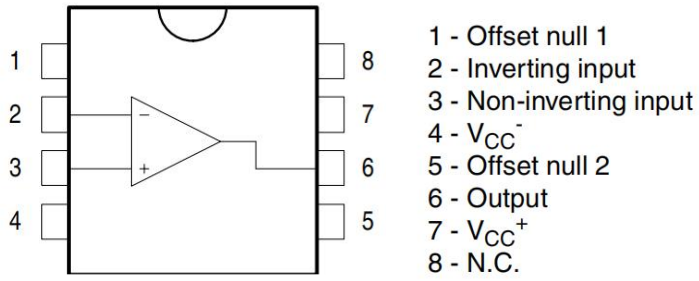

Figure [31]: Schéma de répartitions des broches de l'AOP.

Résultats

Le schéma de câblage incluant la carte Arduino est visible en figure 32. Sur broche de sortie du PWM à 50kHz de la carte Arduino nous tirons un fils (vert) qui est raccordé sur l'entrée de l'AOP. Ce dernier est alimenté par une tension de +15V (rouge) et -15V (bleu). Le signal de sortie amplifié correspond au fils orange, c'est cette tension qui va nous permettre de commander le hacheur.

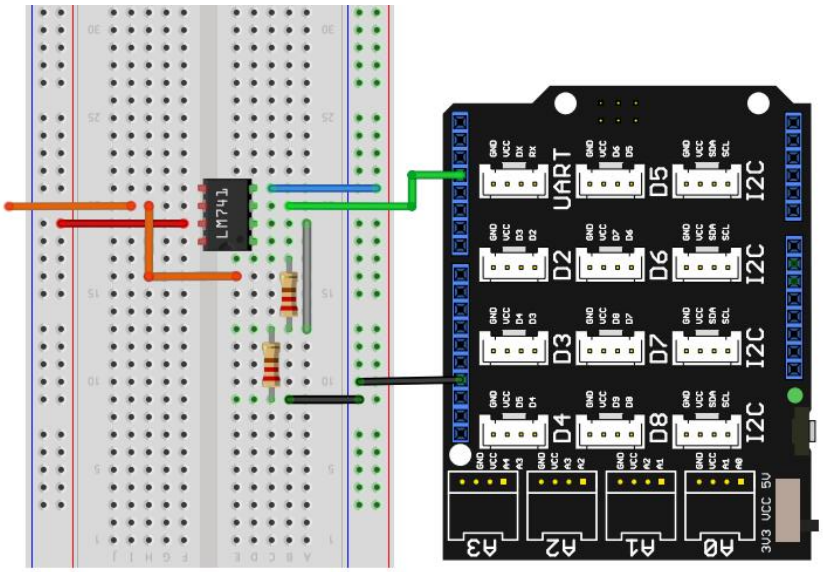

Figure [32]: Schéma de câblage de l'amplificateur non inverseur

Conclusion

Une fois le câblage réalisé et la carte Arduino raccordée à l'amplificateur, nous avons simulé un signal périodique d'une amplitude de 10V, afin de vérifier que l'amplification se fasse correctement, peut importe la valeur de la tension d'entrée. Nous pouvons voir qu'il y a bien une amplification de la tension, sans changement au niveau de la fréquence. Comme en figure 33.

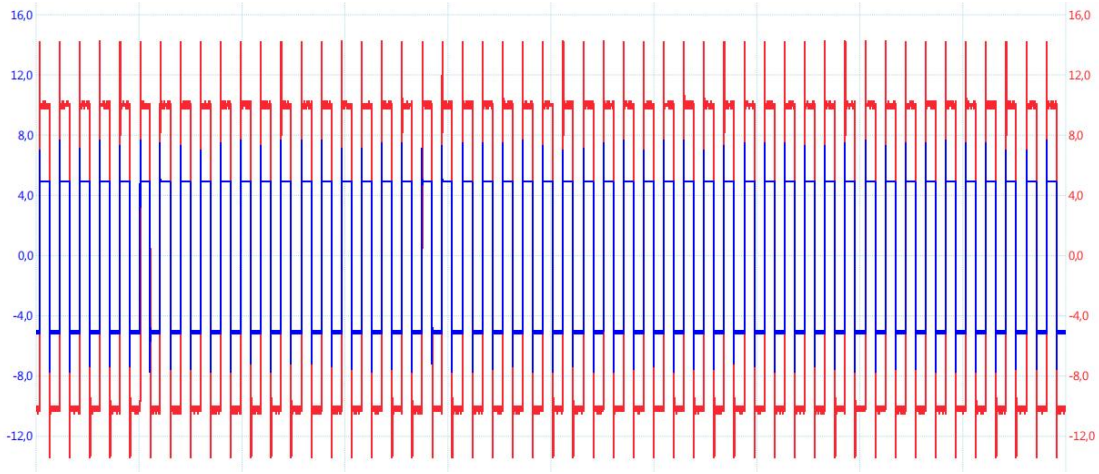

Figure [33]: Signal du PWM amplifié avec un gain de 2

MPPT

Le Panneau Solaire va générer une tension et un courant selon l'ensoleillement extérieur, plus l'ensoleillement sera élevé plus la tension et le courant seront élevés. Avec un ensoleillement constant, pour avoir la tension maximal on doit être en circuit ouvert et alors le courant sera nulle et si on veut un courant maximale, il faut être en court-circuit et alors ce sera la tension qui sera nulle, le point où la tension et le courant sont les plus élevés est appelé Point de Puissance Maximale.
Le MPPT signifie Maximum Power Point Tracker, en d'autre terme la Recherche du Point de Puissance Maximale, c'est ce qui va nous intéresser dans cette partie.

Methode

Pour réaliser le programme du MPPT nous allons devoir suivre son algorithme de fonctionnement visible en figure 34.

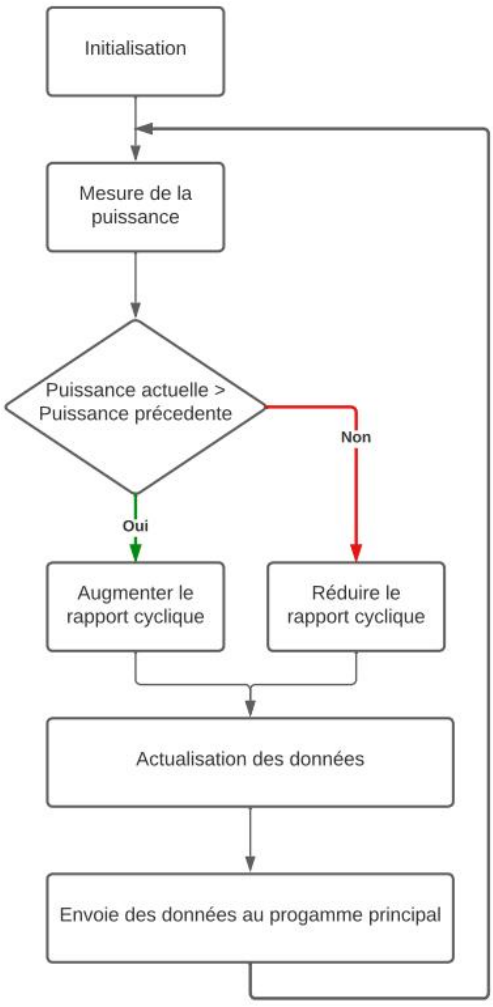

Figure [34]: Algorithme permettant de calculer le MPPT

En regardant l'algorithme nous pouvons voire que lorsque la puissance augmente le rapport cyclique augmente aussi ce qui augmentera la puissance de sortie, tandis que lorsque le panneau solaire va générer moins de puissance; le rapport cyclique va se réduire pour réussir à obtenir le maximum de puissance possible. La gestion de la recherche du point de fonctionnement ce fera dans le programme grâce à une fonction externe que l'on appellera dans le programme principale.

Programmme

Dans cette fonction externe au programme principe on retrouvera tout le fonctionnement de l'organigramme. Pour appeler cette fonction, il faut lui envoyer la valeur de la tension et du courant afin qu'elle puisse recalculer la puissance instantanée. Par la suite cette puissance va être comparée à la puissance précédente pour savoir quelle est la modification qu'il faut apporter au rapport de cycle afin d'avoir la puissance de sortie maximale.
Cette fonction a aussi comme paramètre le rapport de cycle car on va changer sa valeur dans cette fonction. Il correspond au rapport de cycle en pourcentage que l'on veux donner à notre à notre hacheur Elle prendra les valeurs de 10 à 90 de manière à ni atteint le point de court-circuit ni celui du circuit ouvert. De ce fait le paramétrage se fait comme en figure 35.

```
MPPT( courant, tension,&pInst, &PWM);
```

Figure [35]: Appel de la fonction MPPT

Dans cette fonction nous allons calculer la valeur de la Puissance Instantanée (pInst) grâce à la valeur du courant et de la tension, une fois cette valeur obtenue, nous allons la comparer avec la valeur de la Puissance Précédente (pPrec). Comme il est visible en figure 36

```
float pPrec;
*pInst  = tension * courant;
```

Figure [36] : Calcule de la puissance instantanée

Pour calculer la puissance instantanée nous effectuant un simple produit entre lee tension et le courant.

```
if (*pInst > pPrec){
  *PWM = *PWM + 5;
  if(*PWM > 90){
    *PWM = 90;
  }
}
else if (*pInst < pPrec) ,
{
  *PWM = *PWM - 5;
  if(*PWM < 10){
    *PWM = 10;
  }
}
else
*PWM = *PWM;

pPrec = *pInst;
```

Figure [37] : Modification du rapport de cycle en fonction de la puissance

En figure 37 nous pouvons voir la modification du rapport de cycle dans le programme en fonction de la puissance. Il y a deux cas possible, si la puissance instantanée est plus grande que la précédente, cela veux dire que l'on a une augmentation de la puissance et donc nous augmentons la valeur du PWM de 5. Au contraire, si la puissance instantanée est plus faible que la puissance précédente nous avons une diminution de la puissance et donc nous allons réduire la valeur du PWM de 1. Nous prenons tout de même le soins de mettre des limite au valeurs que peut prendre le pwm, de manière à ce qu'il ne soit jamais nul ou jamais à cent pour cent.
Et on fini par actualiser la valeur de la puissance en attribuant la valeur de la puissance instantané à la puissance précédente.

Résultats

Pour le câblage, nous mettons en place le capteur de courant en amont suivit du pont diviseur de tension comme ça la carte Arduino pourra lire, la tension et le courant pour en déduire la puissance. En sortie de la pin 3 de la carte nous mettons en place l'amplificateur de tension pour la commande du hacheur. Nous raccordons les masses pour ne pas avoir de problèmes de mesure. Nous pouvons donc obtenir la figure 38 suivante.

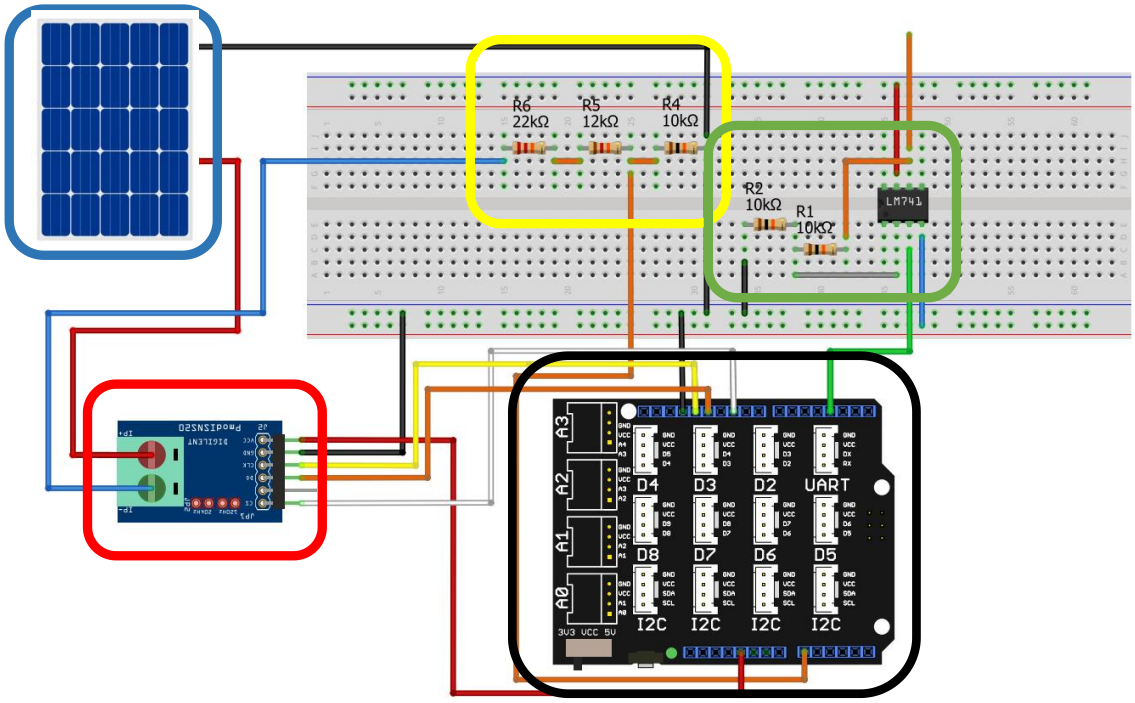

Figure [38]: Schéma de câblage pour la mesure du MPPT

Pour raccorder le panneaux solaire afin de faire les test nous mettons en place le câblage du hacheur série de la figure 38 afin de pouvoir tracer la courbe de puissance.

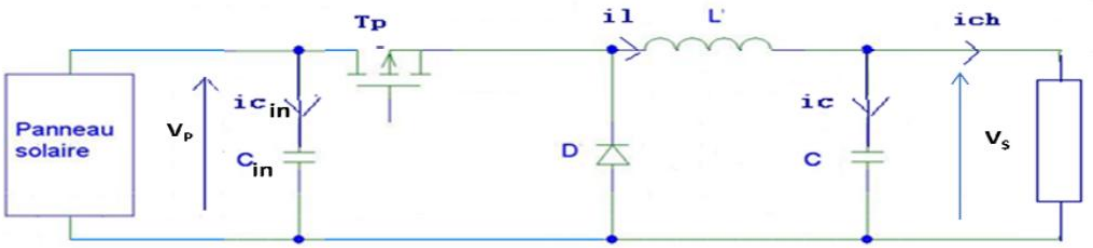

Figure [39] : Montage d'un hacheur série

Une fois le raccordement fait et le programme téléverser sur le micro-contrôleur, on peut commencer à faire les mesures à l'oscilloscope. En branchant sur le port Y la puissance et sur le port X la tension nous obtenons le tracé de la figure 40.

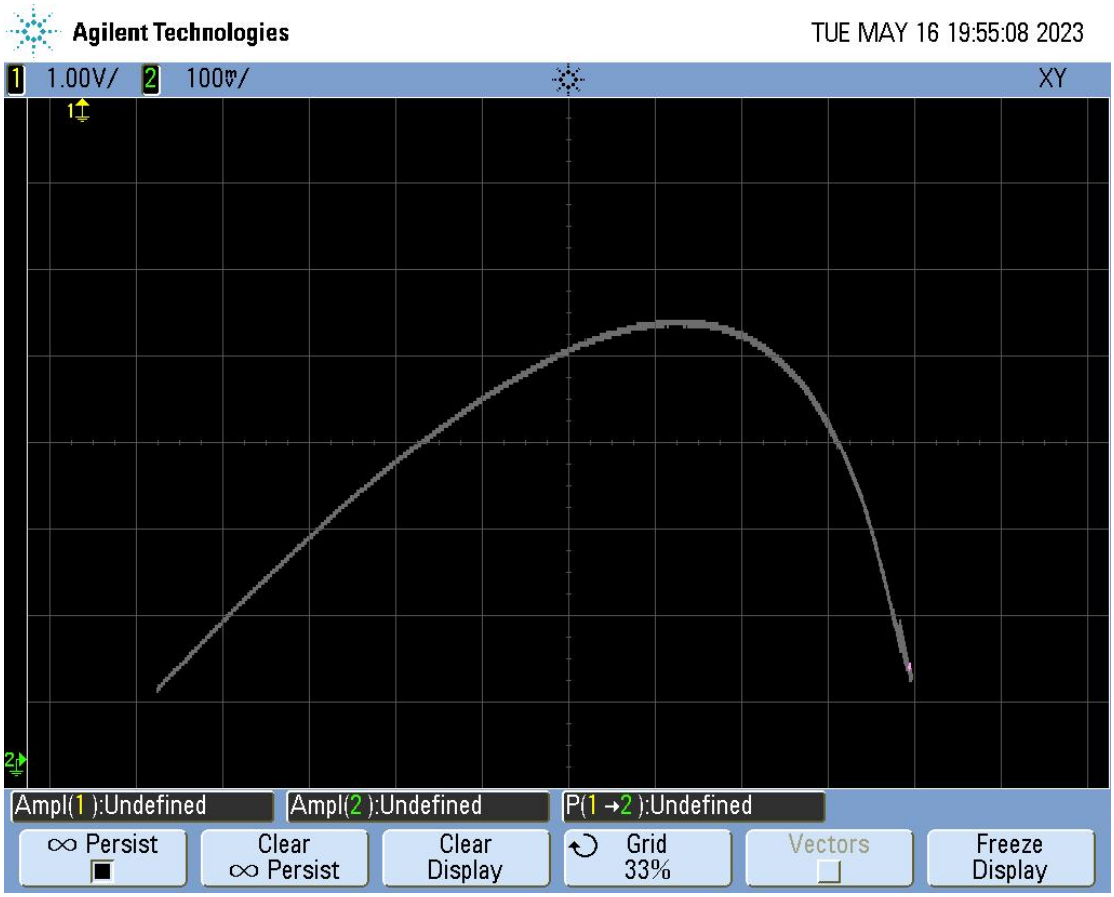

Figure [40] : Tracé de la puissance en fonction de la tension

En circuit ouvert nous avons une tension maximale mais aucune puissance car le courant est alors nul, lorsque l'on augmente la valeur de la charge, la tension diminue mais le courant augmente et l'on obtient alors une puissance qui est croissante jusqu'à un maximum. Si l'on continu, on se rapproche du point de court-circuit et alors la tension diminue et on pert de la puissance.

Annexe

Librairies utilisées :

- SPI.H : Librairie officiell de Arduino permettant d'utiliser le communication SPI, dans notre cas nous l'avons utiliser pour le capteur de courant

- lcd_rgb : C'est une librairie supplémentaire de Arduino qui permet d'utiliser l'afficheur LCD, elle est aussi fonctionnelle si afficheur à un rétroéclairage avec différente couleurs

Sources

[1] Arduino, «Arduino Uno R3», Arduino, 2023
[2] SeeedStudio, «Base Shield V2», Arduino, 2014
[3] Gotronic, «Manuel d'utilisation de l'afficheur I2C LCD 16x2», Joy-IC,2017
[4] DIGILENT , «PmodISNS20 Reference Manual», Texas Instrument, 2016
[5] Locoduino, «Les Timers (IV)», Internet :https://www.locoduino.org/spip.php?article119, [25/05/2023]
[6] Atmel Corporation, «ATmega328P Datasheet», Atmel, 2015
[7] Texas Instruments Incorporated, «TL07xx Low-Noise FET-Input Operational Amplifiers», Texas Instruments,2014